10. 配列（Array）  
・Javaの配列は、同じ型の要素が連続して格納されたデータ構造です。Javaの配列は固定長で、宣言時に配列のサイズを指定する必要があります。配列は、整数、文字、浮動小数点数、オブジェクトなどの様々な型の要素を格納できます。

以下は、int型の配列の宣言と初期化の例です。

int[] numbers = new int[5]; // 長さが5のint型の配列を作成

numbers[0] = 1; // 最初の要素に1を代入

numbers[1] = 2; // 2番目の要素に2を代入

numbers[2] = 3; // 3番目の要素に3を代入

numbers[3] = 4; // 4番目の要素に4を代入

numbers[4] = 5; // 最後の要素に5を代入

Javaの配列は、forループや拡張forループを使用して簡単に操作できま

例）

// 整数型の配列を宣言し、要素数10を指定

int[] myArray = new int[10];

// 文字列型の配列を宣言し、要素数5を指定して初期化

String[] fruits = {"apple", "banana", "orange", "grape", "kiwi"};

配列の要素には、インデックス番号を使ってアクセスします。Javaでは、配列のインデックスは0から始まります

// myArray配列の0番目の要素に10を代入

myArray[0] = 10;

// fruits配列の3番目の要素を出力

System.out.println(fruits[3]); // "grape"が出力される

// fruits配列の要素を順番に出力

for (String fruit : fruits) {

System.out.println(fruit);

}

・JAVAの配列関連METHOD

1. **Arrays.toString()**: 配列の要素を文字列として出力します。
2. **Arrays.sort()**: 配列の要素を昇順にソートします。
3. **Arrays.binarySearch()**: ソートされた配列内で要素を検索します。
4. **Arrays.copyOf()**: 配列のコピーを作成します。
5. **Arrays.fill()**: 配列の全要素に指定した値を設定します。

例）

int[] numbers = {1, 2, 3, 4, 5};

int length = numbers.length;

System.out.println("配列の要素数: " + length);

　例えば、**Arrays.toString()**を使用して、整数配列を文字列として出力するには、次のようにします。

int[] arr = {1, 2, 3, 4, 5};

String arrStr = Arrays.toString(arr);

System.out.println(arrStr);

　　結果：[1, 2, 3, 4, 5]

同様に、**Arrays.sort()**を使用して、配列をソートするには、次のようにします。

int[] arr = {5, 2, 3, 1, 4};

Arrays.sort(arr);

**Arrays.binarySearch()**を使用して、ソートされた配列内で要素を検索するには、次のようにします。

int[] arr = {1, 2, 3, 4, 5};

int index = Arrays.binarySearch(arr, 3);

System.out.println(index);

**Arrays.copyOf()**を使用して、配列のコピーを作成するには、次のようにします

int[] arr = {1, 2, 3, 4, 5};

int[] copy = Arrays.copyOf(arr, arr.length);

**Arrays.fill()**を使用して、配列の全要素に指定した値を設定するには、次のようにします

int[] arr = new int[5];

Arrays.fill(arr, 1);

これにより、**arr**配列のすべての要素が1に設定されます。

**equals**メソッド: 2つの配列が等しいかどうかを比較します。

int[] arr1 = {1, 2, 3};

int[] arr2 = {1, 2, 3};

boolean equal = Arrays.equals(arr1, arr2); // equalにはtrueが代入される